Emille, the Journal of the Korean Electro-Acoustic Music Society Volume 17

한국전자음악협회 학술지 에밀레 제17권

본 학술지는 한국문화예술위원회의 지원으로 제작되었습니다.
아울러 서울대학교 예술과학센터의 협력에 감사드립니다.

The publication of this journal is sponsored by the Arts Council Korea,
with the cooperation of the Center for Arts and Technologies at Seoul National University,

한국문화예술위원회 Arts Council Korea
CAT SNU

Imprint 출판 정보

Editor in Chief 편집장
Lee, Donoung 이돈응
Professor of Composition at Seoul National University 서울대학교 음악대학 작곡과 교수 (작곡, 컴퓨터음악)

Executive Board 운영 위원
Cho, Jinok 조진옥
Jang, Daehoon 장대훈
Jun, Hyunsuk 전현석
Kim, Jonghyun 김종현
Kim, Taehi 김태희
Nam, Sangbong 남상봉
Song, Hyangsook 송향숙
Oh, Yemin 오예민
Yoo, Taesun 유태선

Editor 편집인
Cho, Youngmi 조영미
Lecturer of *EAM* & *MTh* at Suwon University, Cheonnam University etc. 수원대학교 및 전남대학교 시간강사 (컴퓨터음악, 음악이론)

Editorial Board 편집 위원
Chang, Jaeho 장재호
Professor of Music Technology at Korea National University of Arts 한국예술종합학교 음악원 음악테크놀로지과 교수 (작곡, 컴퓨터음악)
Dudas, Richard 리차드 두다스
Professor of Composition and *EAM* at Hanyang University 한양대학교 음악대학 작곡과 교수 (작곡, 컴퓨터음악)
Kang, Joong Hoon 강중훈
Lecturer of Composition and Computer Music at Yonsei University etc. 연세대학교 음악대학 작곡과 시간강사 (작곡, 컴퓨터음악)
Kim, Bumki 김범기
Professor of Composition, Music Education at Gyeongsang National University 경상대학교 음악교육과 교수 (작곡)
Kim, Han Shin 김한신
Lecturer of Computer Music at Seoul Institute of the Arts 서울예술대학교 시간강사 (컴퓨터음악)
Kim, Heera 김희라
Professor of Composition at Kyung-Hee University 경희대학교 작곡과 교수
Kim, Jin-Ho 김진호
Professor of Composition, *EAM* and Musicology at Andong National University 국립안동대학교 교수 (작곡, 컴퓨터음악, 음악학)
Kim, Jun 김준
Professor of Musical Arts and Technology, Dongguk University 동국대학교 영상대학원 멀티미디어학과 교수 (컴퓨터음악)
Lee, Byung-moo 이병무
Professor of Computer Music at Korea National University of Arts 한국예술종합학교 음악원 작곡전공 교수 (작곡, 컴퓨터음악)
Lee, Gi Nyoung 이기녕
Professor of Composition at Dong-eui University 동의대학교 작곡과 교수
Lindborg, PerMagnus 퍼마그너스 린드보르그
Professor of Composition and *EAM* at Seoul National University 서울대학교 음악대학 작곡과 교수
Nam, Unjung 남언정
Professor of *EAM* at Baekseok Arts University 백석예술대학교 음악학부 교수 (컴퓨터음악)
Park, Joo Won 박주원
Community College of Philadelphia 필라델피아 지역전문대학교
Park, Tae Hong 박태홍
Professor of Music Composition and Technology at New York University 뉴욕대학교 (컴퓨터음악)
Parks, Kevin 케빈 파크스 (박케빈)
Professor of Composition and *EAM* at the Catholic University of Daegu 대구가톨릭대학교 음악대학 작곡과 교수 (작곡, 컴퓨터음악)
Shin, Seongah 신성아
Professor of Composition at Keimyung University 계명대학교 음악공연예술대학 작곡과 교수 (작곡)

Advisory Board 자문 위원
Ahn, Doo-jin 안두진
Professor of Composition at Hanseo University 한서대학교 작곡과 교수 (실용음악)
Hwang, Sung Ho 황성호
Professor of Composition at Korea National University of Arts 한국예술종합학교 음악원 작곡과 교수 (작곡, 컴퓨터음악)
Moon, Seong-Joon 문성준
Professor of Composition at Chugye University for the arts 추계예술대학교 작곡과 교수 (작곡, 컴퓨터음악)
Lymn, Young-Mee 임영미
Lecturer of Electro-acoustic Music at Hanyang University etc. 한양대학교 강사 (컴퓨터음악)

Cover design by Kim, Mi-Kyung 표지 도안: 김미경
Issued on 28 December, 2019 발행일: 2019년 12월 28일
Published by Lee, Donoung 발행인: 이돈응
Printed by Yesol Publishing http://www.yesolpress.com 발행처: 예솔출판사 [등록: 제2002-000080호(2002.3.21)]
Dongwoo 4F, 9-24 Yanghwaro6gil, Mapo-gu, Seoul 04044 서울시 마포구 양화로6길 9-24 동우빌딩4층 (04044)

ISSN no.: 2233-9302 국제 표준 정기 간행물 번호: 2233-9302
Price: 28,000 KRW 가격: 28,000원

Using a Probability Distribution Follower in MAS works

Fernando Egido
Independent Composer
Busevin [at] gmail.com
http://busevin.art

In his paper I propose a formal definition of the concept of Probability Distribution Follower and a simple method, based on Monte Carlo Simulation Techniques, to implement it for discrete numbers. I will define a Probability Distribution Follower as a real-time random number generator that is able to fit any change in non-standard Probability Distribution over time. I will show the utility and the possibilities of integrating this algorithm in technological artistic works. Finally, I will explain how this algorithm was used in the work entitled "Transcognition". In this work, I consider that a Cognitive Agent is collaborative when it follows the Probability Distribution of a database that stores all the notes created by the rest of the Cognitive Agents. The Probability Distribution of this database changes over time in an unpredictable way so this algorithm provides the way to follow the changes in real-time regardless of its mathematical definition.

Keywords: Multiple cognitive agent, Algorithmic composition, Aleatory music, Monte Carlo Techniques.

Using random numbers has been one of the key points in generating algorithm music. This is the reason why many different algorithms to create random numbers have been described and used in music. From the Aleatory works of Cage to the stochastics works of Xenakis, many forms of music have used a wide range of different forms of using and generating random numbers. One of the most significative definitions of random number generators is their Probability Distribution .

Random numbers are a very important topic in Computer Science. A random number generator is considered good when we cannot predict any pattern in the numbers and they fit a uniform distribution. The mathematical definition of random numbers and the metrics proposed to measure them are beyond the scope of this paper. I recommend (Downey / Hirschfeldt 2010) for further insight into this topic.

When I was composing *Transcognition,* I realized that I needed an algorithm that was able to produce random numbers, which followed real-time changing Probability Distribution. This was the reason why I implemented this easy algorithm.

Computer Music literata provides several algorithms that are able to generate random numbers that fit a Probability Distribution. For example, we can find an example of an algorithm that fits a Gaussian Distribution in the classical book (Dodge / Jersey 1985). Gaussian or normal Distribution is one of the most used Probability Distributions in Computer Music and Digital Art.

The function that defines the Density Function of Gaussian Distribution is:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \qquad (1)$$

And the algorithm that they proposed (written in C + +) to generate random numbers that fit this Probability Distribution is (Dodge / Jersey 1985: 349)

```
Float gauss (float sigma, float mu)
{
        Float fran ();
        Int k;
        Int const N=12;
        Float const halfN=6;
        Float const scale=1;
                Float sum=0;
                For (K=1; k <= N; k++)
                Sum+=fran ();
                Return sigma*scale*(sum-halfN) +mu;
}
```

In this and other similar books we can find many algorithms that generate random numbers according to Probability Distributions that can be defined mathematically via a Density Function.

There are Probability Distributions that have a known Density Function, but there are many others that, at least apparently, do not fit any of these standard Density Functions.

In artistic applications, we sometimes need to generate random numbers according to an unknown Density Function. For example, we might want to create a work in which, instead of following a mathematically defined Probability Distribution, numbers fit an empirically generated one, or the Probability Distribution of a database. For example, a composer might want a random number generator that follows the Probability Distribution of the pitches in the work of Bach. Sometimes this kind of Probability Distributions does not fit any standard or specific Known Density Function so we cannot use a concrete algorithm to produce Random numbers that follow these distributions.

The Monte Carlo Simulation Techniques are a set of tools and techniques that solve problems by generating repeated random numbers. The Monte Carlo Simulation Techniques provide an algorithm that is able to create random numbers that follow a non-Standard Probability Distribution. The only thing we need to do is to create a table that maps an index ***(i, …., j)*** representing an event with the Probability P_i associated with it.

The use of Monte Carlo Simulation Techniques has a long history in musical applications. We can trace its use back to the very first Computer Music work. The *Illiac Suite* used it for the first time in 1956. The piece was composed by Lejaren Hiller and Leonard Isaacson using a Monte Carlo Method **(**Pareyon / Pina-Romero **/** Agustin-Aquino **/** Lluis-Puebla 2017: 244).

The table must be normalized so the sum of all probabilities equals one. The algorithm used by the Monte Carlo Method is the von Neumann rejection technique (Neumann 1951).

For the creation of random numbers according to any Probability Distribution normalized and mapped into a table,we can follow these steps:

1. Generate two random numbers (a_i ,a_p)
2. Obtain the probability of the index of the first random number *p*(a_i).
3. If $a_p \leq p$ (a_i) go to step 4 or else go to step 1.
4. Return a_2 as a valid random number that follows the Probability Distribution.

No matter which Probability Distribution is represented in the table, the numbers that pass step 3 will fit that represented by this table.

But this is not a very efficient algorithm. For example, if we use this algorithm to generate random numbers that follow Gaussian Distribution, we will need to generate over 200 numbers until we find one that passes step number 3 (that satisfies the formal condition to be considered as a valid random number for the purpose of this paper). If we use the Dodge algorithm, we will only need to create 12. Therefore, if we need to create numbers according to a known, mathematically defined Probability Distribution, it is better to use a standard algorithm. Generating random numbers is a complex task that requires computational time, so the fewer random numbers we need to use to create the desired aleatory numbers, the better the algorithm is.

Definition

The proposed definition is: a *Probability Distribution Follower* is an algorithm (or any other thing) that is able to generate real-time random numbers that follow a Changing Probability Distribution over time that does not necessarily have a Known Density Function. The term is inspired by the envelope follower. In the same way, an envelope follower is able to extract a feature from a signal that changes in time and returns the actual value of the followed parameter.

We have a finite number of Probability Distributions that have a discrete and finite number of states represented in the set ***S***. Each one of these states represents a concrete form of the mapped and normalized table. We have a time series (set ***T***) each one representing the moment in which the random numbers are called.

Formal definition

We can define a *Probability Distribution Follower* as a structure composed of the elements below:

Set of events *E= {e_i, …, e_j}*

Set of index *I= {i, …, j}*

Set of probabilities of events *P= {p_i, …, p_j}*

Set of discrete random numbers *A= {a_1, …, a_n}*

Set of pairs of random numbers

PA= {$(a_i \in A \wedge a_p \in A)_{1,\ldots,}$ $(a_i \in A \wedge a_p \in A)_n$}

Set of times *T= {t_1, …, t_n}*

Set of states of Probability Distributions at $t \in T$

S = {s_{t_i}, …, s_{t_n}}

Set of applications *PS:* $I \rightarrow P$ that relates one of the indexes $i_i \in I$ with one of the $p_i \in P$ *in the states* $s_t \in S$ *PS= {ps_{t_i}, …, ps_{t_n}}.*

We formally define a *Probability Distribution Follower* as an algorithm (or any other thing) that is able to generate pairs of random numbers $\{a_i \in A \wedge a_p \in A\} \in PA$ at the time $t \in T$ that satisfies the condition:

$$a_p \leq ps_t(a_i) \qquad (2)$$

Implementation

One algorithm that implements this can be the following (in pseudocode):

1. Begin the clock and wait until the first $t_i \in T$comes.
2. When t_i arrives obtain the state s_t with the application $ts_i$and store it in the buffer.
3. Generate the table ps_t from $ts_i$and store it in a buffer.
4. Generate two random numbers. (a_i ,a_p)
5. Consult in the table ps_t the probability of occurrence of the index a_i (pts_{a_i}).
6. If $a_p > pts_{a_i}$ go to step four and generate another pair of random numbers otherwise go to step 7.
7. If $a_p \leq pts_{a_i}$Then return a_p as a valid random number, *i=i++.*
8. If *i=n* then go to step 10.
9. Delete the buffers and go to 2 to generate the next number.
10. Finish.

Obviously, this algorithm can be used in constant or non-constant time intervals. For example, in the work entitled *Transcognition,* the Cognitive Agent uses the duration of the notes as a time interval.

If we all wanted, we could create a timeframe to reset the database to the initial state in such a way that instead of using the total database of the numbers shown, we would only use a reduced timeframe. So, we would generate numbers that follow only the Distribution Probability followed by the numbers in this timeframe or temporal window.

Using Probability Distribution Follower in the work entitled *Transcognition*[1]

Work Description

Transcognition is a work composed in real-time that has a different function each time that the Max/Msp patch is launched. It is an open-form work created algorithmically. The audience will see the patch in the video (in *figure 1* you can see how the patch looks during the performance), the main window of the patch during the creation of the work. The patch has been programmed completely by the composer, without libraries or external sources.

The title of this work is a concept created by Graeme L. Sullivan (Sullivan 2010). This concept was created as a thinking artist model. It describes artistic cognition as something that is very dynamic and changes very often, in which meanings change recursively in the very process. The way in which the artist creates the work makes the perceived environment mutable and therefore the cognition of the work of art is transformed by the creative process itself.

> Transcognition is a visual arts process that knows that the forms, ideas, and situations are informing agents of the mind that surrounds the artistic self during the practice of visual arts. Transcognition is what happens when a creative process, which is closely related to a changing environment, changes itself in the same creative process. (Sullivan 2010: 130)

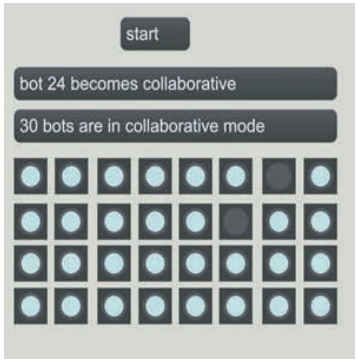

Figure 1. The main window of the Max/Msp Patch that shows which Agents are in collaborative mode.

This piece is created by 32 Multimodal Multiple Cognitive Agents (following the MAS paradigm (Wooldridge 2009). These agents have the capacity of creating music in competitive or collaborative modes (that is why the work is called *Transcognition*; at the same time, the environment changes in a constant evolving relationship). The work begins with all the agents in competitive mode. Each Agent creates its own music trying to generate the most interesting one to get the audience focused on its activity.

This transcognitive process also makes the score a negotiation space among all Cognitive Agents. Each Agent produces a single voice from a traditional perspective, creating a 32-voice polyphony.

In this work, competitive or collaborative modes are defined technically by following a different non-standard Probability Distribution that changes continually.

The Work uses two different tables for each parameter, where the two different Distribution Probabilities are represented. The competitive table can be considered the inverse of the collaborative one. In figure 2 you can see the main sub-patch of the work in which all the Agents use the collaborative or competitive table, depending on their working mode, to generate musical events according to their internal state (collaborative or competitive).

When they are in collaborative mode, Max/Msp stores each event occurring in a table. (This table is normalized in an auxiliary table). This table stores all the events that every cognitive agent has created to date. This database with all the event occurrences is converted into a Probability Distribution by normalizing it each time that a new event arrives. In this way, all the Agents that are working collaboratively tend to repeat the most frequent notes.

When they are in the competitive mode, they will use another table in which the probabilities are inverted so the index most likely to appear in the cooperative mode now has less probability. In other words, they will generate notes that are less used.

This follows an interpretation of the Shannon Theory of communication (Shannon 1948), according to which the numbers least likely to occur are more significant.

In this way, when the Agents are in competitive mode, they will try to generate the most significant note, trying to get the focus of the audience, while when they are in the collaborative mode, they will generate notes that are not significant but are the result of an implicit negotiation with the other Agents.

The result of the progressive change of the work is a special narrative that arises from the fact that the Agents slowly enter the collaborative mode. So, while several Agents are negotiating, others still try to get the focus of the audience, but this number gradually decreases. At the end of the work, all the Agents are in the collaborative mode so the work slowly changes creating a sort of continuous variation. At the same time, the Cognitive Agent

uses the score of the work itself as the vision of their reality. So, they interact with this micro-world that is created music. The agents see reality through the score of the work as a database while, at the same time, they react to this score and recursively the score changes according to the Agents' reaction. In this way, we have constant interaction between reality and the way the Cognitive Agents see it. That is the definition of transcognition. Agents are transformed by the very fact of creating the score of the work.

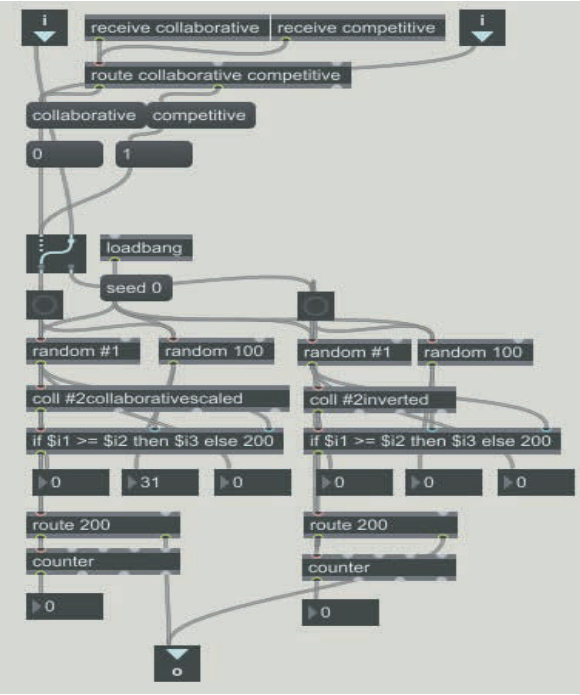

Figure 2. This sub-patch called *followdistr* implements in Max/Msp the described algorithm. If the Cognitive Agent is in collaborative mode, it uses the table *collaborativescaled* but if it is in competitive mode, it follows the table *inverted*. Those tables are shared by all Cognitive Agents.

At the beginning of the work, all Cognitive Agents give their first note. The tables begin in a Uniform Distribution. Each Cognitive Agent has 6 sub-patches *followdistr,* each one related to a specific parameter of the work (duration, pitch, dynamic, timbre, envelope, spatial position). That is why we have two tables for each parameter (you can see this in *figure 3*). Every time a note finishes, a new one is created.

Using Probability Distribution Follower

For creating the database, each time that a Cognitive Agent generates a note it is stored in 6 different tables that form the overall database. In each one, a concrete parameter is stored.

event	index	Number of occurrences
pp	1	1
p	2	2
mp	3	17
mf	4	6
f	5	1
ff	6	12

Table 1. Database that stores the number of occurrences for each dynamic considered in the work.

This table is normalized and converted into a Probability Distribution with the function.

$$P_i = \frac{X_i}{\sum_{i=1}^{j} X_i} \qquad (3)$$

In *table 2* you can see *table 1* normalized.

index	Probability
1	0.0256
2	0.0512
3	0.4359
4	0.1538
5	0.0256
6	0.3073

Table 2. Probability Distribution of dynamics. This is the table used in collaborative mode *collaborativescaled*.

To create the competitive table, *table 1* is normalized with this function (probabilities are inverted). This table will be used in competitive mode. Other forms of converting *table 1* into this were considered, but the resulting sound from this one was the best. This is, of course, arbitrary, and there is no specific or unique grounds for this.

Max value of the table =*mv*

$$P_i = \frac{(mv - X_i)}{\sum_{i=1}^{j} (mv - X_i)} \qquad (4)$$

index	Probability
1	0.2284
2	0.2142
3	0
4	0.1570
5	0.2284
6	0.1714

Table 3. Database that describes the Probability Distribution for the competitive mode *inverted*.

Changing to collaborative

Each one of the Cognitive Agents believes that it is participating in a cooperative game to create music. They act as listening machines that have the possibility to listen and react to the overall score choosing whether they work in the competitive or in the collaborative mode. They are supposed to be intelligent enough to choose in which mode they will perform to create a better result.

Periodically, each agent checks whether its work is really the most significant. If they decide not to, they become collaborative. Cognitive Agents have a metric based on the Theory of Communication (Shannon 1948), to measure significance, comparing their own part with the overall score. After several negative measurements, they become collaborative. Each *agent* object is instantiated with the agent number, the metric sensibility for mode changing and the frequency of the evaluation of the mode change.

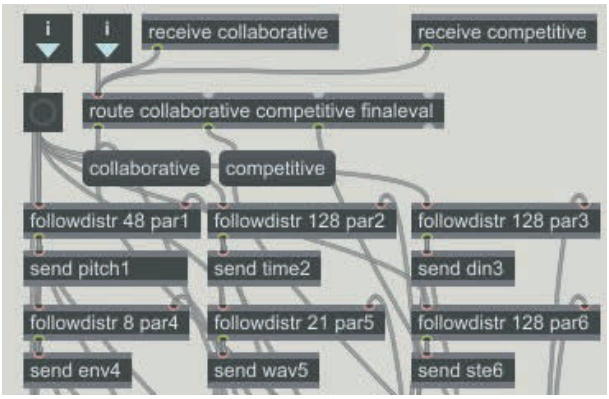

Figure 3. This part of the sub-patch, called *agent* that implements the Cognitive Agents shows the *followdistr* sub-patches. The *followdistr* sub-patches are instantiated with the cardinality of the *E* set and the number of the corresponding parameter (*par1* to *par 6*).

This uses a paradox. If each Cognitive Agent tries to get the Focus, the result is that none of them can actually get it, because the real result is very close to a Uniform Distribution in which any note is more significant. To do this, each Cognitive Agent periodically compares its own database with the general database of the work (their own voice with the general score) and if it thinks that what it is doing is significant, it continues in competitive mode, otherwise, it will enter the collaborative mode. Therefore, if the work is to become significant, the agent had better become collaborative.

Finally, all Agents become collaborative. Two minutes after the last Agent becomes collaborative, the work finishes automatically. Once they become collaborative, they cannot return to the competitive mode.

The input or contribution of the Cognitive Agent is the score. In this way, the generated score is converted into the environment through which the Agents relate to each other. The role of the score itself is completely transformed in this work. Instead of being something closed that all Agents must follow, it is also the source of information that the Agents use to create their own music. The contribution of the Agents is exclusively the score (the dataset of all the notes produced by all the Agents) that converts them into listening machines. The inner state of the Agents is changed by the score and, at the same time, they change and participate in it.

Instead of a classical form, we have an open process form in which the sounding material changes according to the mode changing of the Agents. The work is constantly changing in an unpredictable way because the database (the score) changes completely according to whether the Agents become collaborative, so they are constantly imitating a different thing at all times.

Conclusion

Research in using random numbers in Computer Music works is still an open field that can arouse the composers' imagination to provide new compositional strategies. Furthermore, the modelization of social agents into computational Cognitive Agents permits a new form of sonification of social Processes.

References

Dodge, C. / Jersey, T. A. (1985). *Computer Music: synthesis, composition, and performance*. New York: Schirmer.

Downey, R. G. / Hirschfeldt, D. R. (2010). *Algorithmic Randomness, and Complexity.* Berlin: Springer-verlag.

Neumann, J. V. (1951). "Various techniques in connection with random digits". *Monte Carlo Method. Applied Mathematic series National bureau of standards* 12/1: 36–38. Washigton D.C.: US Government Prionting Office.

Pareyon, G. / Pina-Romero, S. / Agustin-Aquino, O.A. / Lluis-Puebla E. (2017). *The Musical-Mathematical Mind.* Berlin: Springer-verlag.

Shannon, C. E. (1948). "A mathematical theory of communication". *The Bell System Technical Journal* 27/3: 379 - 423.

Sullivan, G. (2011). "Artistic cognition and creativity". *The Routledge Companion to Research in the Arts*. Michael, B./ Henrik, K. [Eds.]. Oxford and New York: Routledge.

Sullivan, G. (2001). "Artistic Thinking as Transcognitive Practice. A Reconciliation of the Process-Product Dichotomy". *Visual Arts Research* 27/1: 2–12.

Sullivan, G. (2010). *Art Practice as Research: Inquiry in Visual Arts*. Thousand Oaks:Sage.

Wooldridge M. (2009). *An Introduction to Multi Agent Systems*. John Wiley and Sons.

[1] https://busevin.art/transcognition

[Abstract in Korean | 국문 요약]
멀티에이전트시스템MAS에서 확률 분포 추적자 사용하기

페르난도 에기도

이 논문에서는 확률 분포 추적자 Probability Distribution Follower의 개념을 정의하고 몬테 카를로 시뮬레이션 기법에 기반한 간단한 방법을 제안하며, 이를 불연속 수로 이행하였다. 저자는 확률 분포 추적자를 실시간 랜덤 숫자 생성자로 전제하고 시간 흐름에 따라 비표준적인 확률 분포의 변화에 적용될 수 있도록 하였다. 기술을 활용한 예술작품에서 이러한 알고리듬을 통한 유용함과 가능성을 보여줄 것이다. 끝으로, 저자는 "인식 Transcognition" 이라는 제목의 작품에서 이 알고리듬이 어떻게 사용되었는지 설명한다. 이 작품에서, 저자는 하나의 인지 에이전트가 다른 나머지 인지 에이전트들이 생성한 모든 음을 저장한 데이터베이스의 확률 분포를 뒤따를 때 협력한 collaborative 것으로 간주한다. 이 데이터베이스의 확률 분포는 시간에 따라 예측불가능하게 변화하기 때문에, 알고리듬은 수학적 값에 상관없이 실시간 변화를 따를 수 있는 방법을 제공한다.

주제어: 다중 인지 에이전트, 알고리즘 작곡, 불확정성 음악, 몬테 카를로 테크닉.

논문투고일: 2019년 9월22일
논문심사일: 2019년 11월29일,12월2일
게재확정일: 2019년 12월5일